TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

WEBOVÉ ROZHRANÍ PRO SPRÁVU KURZŮ

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Zdeněk Pohl**
Vedoucí práce: Ing. Zbyněk Mader, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

WEB INTERFACE FOR MANAGEMENT COURSES

Diploma thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technology

Author: **Bc. Zdeněk Pohl**
Supervisor: Ing. Zbyněk Mader, Ph.D.

Liberec 2014

Tento list nahraďte
originálem zadání.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Abstrakt

Cílem práce bylo vytvořit webový portál pro správu kurzů.

Systém disponuje funkcemi pro zavedení a editaci kurzů včetně správy aktualit (přidání, editace, mazání).

Dále jsou dostupné funkce umožňující správu článků. Portál poskytuje grafické prostřední pro zobrazování přehledu přihlášených uživatelů či plateb za kurzy.

Veškeré funkce jsou vázány k uživatelským rolím, které jsou v systému implementovány, tj. administrátor, který disponuje větší škálou použitelných funkcí než registrovaný uživatel. Administrátor má možnost registrované uživatele mazat.

Důležitou částí práce je i samotná grafická podoba portálu, neboť je třeba užít snadno ovladatelný, přehledný a zároveň vzhledově zajímavý portál.

Klíčová slova:

Web, portál, php, semináře, databáze, mysql, jquery, kurzy

Abstract

The aim of my work was to create a web interface for management courses

The system have available functions for implementation and editing of courses including administration of news (inserting, editing, deleting.)

Further there are available the functions, which enable the administration of articles. The portal provide graphical environment for display of overview of logged in users or payments for courses.

All the functions are connected to the user roles, which are implemented in the system, e.g. administrator, who has at his disposal a wider scale of usable functions than registered user. The administrator have the possibility to delete the registered users.

An important part of my work also is the graphical design of the portal itself, because it is necessary to use an easy manageable, clear and at the same time visually interesting portal.

Keywords:

Web, portal, php, seminaries, database, mysql, jquery, courses

Poděkování

Poděkování patří Ing. Zbyňku Maderovi, Ph.D., za možnost vytvořit tento projekt, za skvělé vedení práce a důležité postřehy či rady.

Rád bych poděkoval i Mgr. Petrovi Kubějovi za korekturu práce a opravu chyb.

Obsah

Seznam obrázků

1 Úvod do problematiky

V dnešní době se technologie posouvají výrazně vpřed, zejména technologie internetu a webových stránek. Celý internet je plný portálů, systémů a dalších webových aplikací, které uživatelům poskytují nejen možnost vyhledávání informací prostřednictvím webových vyhledávačů, ale i možnost využít pokročilejší funkce, které v dřívější době – zejména v začátcích tohoto systému – byly nemyslitelné. Jsou to např. funkce umožňující komunikaci s bankovním účtem klienta nebo portály, které poskytují možnost spravovat veškerý obsah daných stránek.

Webový portál je ucelený systém, který návštěvníkům (klientům) poskytuje kompletní služby v konkrétním průmyslovém či jiném odvětví.

Portál pro správu kurzů by měl návštěvníkům podat kompletní informace o pořádaných seminářích, poskytnout možnost se na seminář přihlásit či uchovávat informace o všech navštívených seminářích.

Nejdůležitější částí práce je nastudovat problematiku pořádání kurzů, co je třeba znát k vytvoření seminářů včetně všech doplňkových informací, aby celý webový portál po dokončení byl samostatný, soběstačný, ovládaný pouze administrátorem bez nutnosti znalosti nějakého programovacího jazyka.

Základem portálu jsou semináře, kde u každého z nich je třeba znalosti konkrétních atributů, jako např.:

a) datum semináře – v případě, že je vícedenní, tak kdy seminář začíná a kdy končí,

b) název semináře,

c) cena semináře a jeho možná kapacita,

d) město, ve kterém se seminář bude odehrávat a upřesnění místa v dané destinaci.

Je nutná znalost i dalších položek, jako jsou např. PDF soubory, co by příloha k danému semináři. V něm si může každý návštěvník přečíst, o čem detailněji seminář pojednává.

Všechny tyto parametry, včetně dalších údajů o platbách a uživatelích je třeba ukládat, aby byly v případě potřeby k dispozici.

Pro tuto možnost je využita databáze, která byla navržena tak, aby bylo zachováno vhodné propojení kurzů, účastníků a jejich přihlašování apod.

2 Analýza systému

Na základě požadavků zákazníka byla provedena analýza systému, aby nedocházelo k protichůdným požadavkům a celý systém mohl být vhodně navržen a implementován.

Základní částí analýzy systému je sběr požadavků, což je důležitá část analýzy, neboť popisuje všechny funkce, které je nutné navrhnout a implementovat.

Druhou částí je analýza požadavků, která slouží pro ujasnění neúplných nebo protichůdných požadavků na funkce nebo doplnění chybějících informací nutných k samostatné implementaci funkcí.

2.1 Sběr požadavků

Sběr požadavků je v rámci analýzy nejdůležitější část, neboť bez ní by programátor nevěděl, jaké funkce klient požaduje, jak by měla vypadat daná struktura systému či pro jakou cílovou skupinu je portál určen.

Systém musí obsahovat následující funkce:

1) Rozdělení uživatelských rolí – *Administrátor* a *Uživatel,* kde každá role má definované specifické funkce, které smí používat a jsou mu dostupné.
2) Možnost registrace potenciálního klienta na portál, tj. musí být uživateli umožněno registrování a následně přihlášení na portál.
3) Přihlášenému uživateli musí být systém schopný poskytnout informace o navštívených kurzech, nebo o kurzech, na který je uživatel přihlášen – tzn. každému uživateli poskytnout jeho historii kurzů i kurzy, na které je zapsán včetně stavu platby – zda má uživatel zaplaceno či nikoliv.
4) Kompletní správa seminářů – možnost seminář vytvářet, editovat či mazat. Musí být umožněno kopírování semináře, včetně implementace aktivace/deaktivace semináře, která určí viditelnost pro nepřihlášené uživatele či uživatele, kteří nedisponují právy administrátor.
5) Kompletní správa aktualit – možnost danou aktualitu vytvářet, editovat či smazat.
6) Správa textových informací v konkrétních blocích na stránce – zejména záložky a menu, ve kterém se nachází důležité informace.

7) Poskytnutí administrátorovi ucelený přehled o pořádaných kurzech včetně přihlášených uživatelů na konkrétní kurz a stavu jejich plateb – zda uživatel již zaplatil či nikoliv.

8) Možnost pohledu na stav vlastního bankovního účtu, ke kterému se budou vztahovat všechny platby za pořádané kurzy od klientů včetně možnosti filtrování pohybů na bankovním účtu (zadáním data „od“ a data „do“).

9) Import všech dostupných seminářů z konkrétní webové adresy (kterou využíval klient doposud).

10) Možnost vytvoření, editace či smazání partnerských webových stránek, které se budou zobrazovat ve spodní straně portálu včetně možnosti ve vedlejším bloku při stejném výčtu funkcí spravovat i loga partnerů.

2.2 Analýza požadavků

Analýza požadavků je metoda sloužící ke zjištění nesrovnalostí jak v rámci sběru dat (zda jsou stanoveny všechny funkce a jejich činnost), tak v rámci nekompletních informací. Všechny problematické požadavky je třeba před samotným návrhem systému odladit, prokonzultovat se zákazníkem a navrhnout vhodné alternativní řešení.

Na základě analýzy požadavků a jim předcházejícího sběru dat, byla hlavní část systému oddělena ihned po přihlášení – administrátorské menu. Toto menu by mělo být umístěno na přehledném místě, čitelně označeno včetně logických popisů, aby byl administrátor na základě přečtení např. pouze názvu tlačítka schopen zjistit, jaká funkce to na portálu je. Je tedy nutné dbát na vhodné umístění tohoto menu.

Součástí tohoto menu bude i klasické uživatelské menu, které bude obsahovat pouze konkrétní funkce, které může přihlášený uživatel bez práva *Administrátora* využít.

2.2.1 Uživatelský profil

Blok „Uživatelský profil“ je umístěn v uživatelském menu a bude dostupný pro libovolného přihlášeného uživatele. Na této stránce se budou nacházet aktuální data o uživateli, který je přihlášen – jeho uživatelské jméno, email, pod kterým byl registrován apod.

2.2.2 Vlastní semináře

Funkce, která bude dostupná po kliknutí na konkrétní položku v menu, umožní uživateli zjistit informace, na jaké aktuální semináře je přihlášen, přičemž ke každému semináři přibude i informace o stavu platby včetně možnosti aktuální seminář opustit. Budou zde dostupná i data týkající se navštívených seminářů. Tyto výpisy budou určeny opět pro libovolného přihlášeného uživatele.

2.2.3 Správa článků

Možnost spravovat veškeré články, resp. textové informace, se bude nacházet v administrátorském menu. Správa článku bude skýtat možnosti editace, přidání a mazání článků, přičemž každý článek bude mít přidělen, tzv. „zobrazovací blok“, což je blok, kde se daný článek zobrazí. Sekce správy článku bude dostupná pouze uživateli s rolí administrátora.

2.2.4 Správa seminářů

Administrátor bude mít možnost využít základních funkcí, jako jsou vytvoření semináře, jejich editace či smazání. Mimo jiné, budou dostupné i rozšiřující funkce, zejména pro kopírování semináře, nebo pro zjištění stavu plateb za kurzy.

Všechny dostupné semináře (dostupnost semináře značí stav, kdy je kapacita větší než 0, a zároveň je seminář aktivní) budou vypsány ve výpisu všechny aktuálních kurzů, který bude dostupný všem uživatelům (tedy i těm, kteří nejsou přihlášení, resp. registrováni).

2.2.5 Správa aktualit

Funkce spravující aktuality budou dostupné taktéž pouze administrátorovi, budou mu umožňovat vytvářet, editovat a mazat aktuality – tj. zprávy, které se budou zobrazovat na stránce jako „novinky“ pro portál.

2.2.6 Správa uživatelů

Blok pro uživatele bude poskytovat přehled registrovaných uživatelů na portále s možností konkrétního uživatele smazat. Bude dostupný pouze administrátorovi.

2.2.7 Správa partnerů a odkazů

Spodní část stránky, tzv. „zápatí“, obsahuje sloupce pro partnery a odkazy. Administrátor má možnost vytvořit, editovat nebo smazat partnera nebo odkaz. Rozdíl

mezi těmito dvěma bloky je v tom, že partner bude mít jiné atributy než odkaz, zejména bude rozšířen o logo. Součástí vytvoření nového partnera nebo odkazu bude i nastavení pořadí pro výpis (v jakém pořadí se daný partner či odkaz zobrazí na stránce, přičemž nižší hodnota u pozice značí vyšší pozici – zobrazení k začátku) v konkrétním bloku.

3 Návrh řešení

Tato kapitola má za cíl vytvořit grafickou podobu portálu, která by byla nejen pro uživatele, kteří portál budou navštěvovat, ale i pro administrátora, který ho bude ovládat. Vzhled portálu by měl být graficky přívětivý a přehledný. Pro uživatelskou i administrační část byla využita stejná grafická podoba menu, kde se podle práv zobrazí patřičné funkce. Nebylo tedy třeba tvořit dvě samostatné grafické podoby pro obě části.

Druhou částí této kapitoly je samotný návrh databázového systému, ve kterém je definováno, jaké atributy je třeba uchovávat a proč jsou pro řádnou činnost systému důležité.

3.1 Návrh grafické podoby

Před samotným návrhem funkcí nebo databáze bylo třeba vytvořit návrh grafické podoby a to tak, aby byla uživatelsky přívětivá a aby se v ní dokázal uživatel snadno orientovat.

Pro grafický návrh webového portálu byl použit program Adobe Photoshop, který umožňuje mnoho bitmapového nastavení, jak v oblasti stylů, barev, přechodů, tak i velikostí či různých formátů. Je třeba mít na paměti, že grafická podoba webu je naprosto odlišná než např. grafické logo společnosti, kde se dají použít všechny dostupné funkce. Grafika webu je limitována možností následného kódování.

3.1.1 Návrh veřejné sekce

Veřejná sekce je dostupná všem uživatelům. Je to část portálu, kterou uvidí všichni návštěvníci bez nutnosti přihlášení, skládající se prakticky ze všech základních bloků každého portálu, což jsou bloky pro menu, hlavní banner, obsahovou část a zápatí.

3.1.1.1 Horní menu

Horní menu slouží k základní orientaci na portálu, umožňuje uživateli procházet portál, resp. stránky portálu, zejména stránku úvodní, na které jsou texty, které vytvořil administrátor v rámci konkrétního článku.

V menu se vyskytuje i ikonka směřující na stránku s výpisem všech dostupných seminářů, které se v budoucnu budou konat. Přes poslední odkaz je dostupná stránka s kontaktními údaji o majiteli stránek, resp. portálu.

3.1.1.2 Banner se záložkami

Banner není nic jiného, než velký obrázek, který se zobrazuje prakticky přes celou šířku okna zobrazujícího portál. Důležitým a zároveň jediným prvkem v banneru jsou na jeho spodní části čtyři záložky, které se po rozkliknutí „vyrolují“. Každá z nich obsahuje text, který ji vystihuje. Díky tlačítku umístěnému na spodní části záložky se může návštěvník dostat na stránku, s kompletními textovými informacemi z oblasti konkrétní problematiky dané záložky.

3.1.1.3 Levý blok obsahové části

Pod bannerem se nachází obsahová část, která je opticky oddělena, potažmo rozdělena do tří částí – levá, pravá a střední.

Levý blok obsahové části slouží primárně jako menu – rozcestník, který umožní podobně jako záložky popsané v kapitole 3.1.1.2, po kliknutí na tlačítko, které je dostupné v každé části tohoto bloku, uživateli zobrazit příslušnou stránku.

Na konkrétních stránkách, které potřebují více prostoru – zejména stránky s výpisy kurzů, uživatelů, správy seminářů apod. je tento blok odstraněn a nahrazen jedním blokem, který vzniknul spojením levého bloku a celé střední části.

3.1.1.4 Hlavní obsahová část

Hlavní obsahová část se nachází ve středu obsahové oblasti, kde se vypisují všechny informace různých stránek, na které se po kliknutí na odkaz dostal čtenář, resp. návštěvník portálu. Mohou to být stránky např. s kontaktními údaji, s výpisem aktuálních seminářů nebo úvodní stránka.

3.1.1.5 Pravý blok obsahové části

Pravá část je důležitá, neboť se do této oblasti vypisují důležitá data, zejména všechny aktuality a aktuální připravované kurzy. Pokud je dostupný kurz, který je aktivován, je zobrazen na stránce s výpisem seminářů a pokud je atribut určující zobrazení na hlavní stránce nastaven kladně, zároveň se tento seminář zobrazí i v tomto pravém menu.

Nad výpisem seminářů jsou vypsány i aktuality, které mohou být vytvářeny libovolně, nezávisle na seminářích.

3.1.1.6 Zápatí

Blok zápatí je umístěn ve spodní části stránky, pozadí je zvoleno tmavě šedé, tedy takové, aby bylo barevně kontrastní k barvě obsahové části, která je světle šedá. V tomto bloku jsou dostupné informace o partnerech či spřátelených internetových stránkách, nachází se zde i možnost přihlášení na portál a s neposlední řadě odkazy na sociální sítě jako Facebook nebo Twitter.

3.1.2 Návrh uživatelské sekce

Uživatelská sekce je část stránky, která se zviditelní uživateli, jenž se přihlásil na portál. Podmínkou pro přihlášení je povinná registrace, aby si portál uložil data k tomuto uživateli.

Prakticky se jedná o menu, které se zobrazí pod bannerem v šedém pozadí. Slouží jako rozcestník pro dostupné funkce uživatele, potažmo administrátora.

Uživatelská sekce je totožná s administrační sekcí s rozdílem v počtu zobrazených funkcí. Uživatel, který disponuje právy *administrátor*, má dostupné všechny funkce, naopak pouhý uživatel má k dispozici pouze vybrané funkce.

3.1.3 Návrh administrační sekce

Administrační sekce slouží pro administrátora. Díky této sekci je schopen celý portál ovládat a využívat veškeré dostupné funkce. Sekce má podobu šedého obdélníku, který se zobrazuje vždy pouze po úspěšném přihlášení v roli administrátora. V tomto bloku jsou vypsány možné funkce, které mohou být využity. Celý blok se zobrazuje pod bannerem se záložkami, viz kapitola 3.1.1.2.

3.1.4 Návrh grafických prvků

Grafické prvky na portálu jsou prvky, které mají určité grafické vlastnosti. Tvoří stránku pro uživatele zajímavější svými funkcemi, jako např.:

a) rozklikávací menu,

b) otevírací bloky pro práci s formuláři,

c) informativní hlášky, které po určitém časovém úseku sami zmizí.

Tyto grafické prvky dokáží uživateli ulehčit práci např. v rámci správy seminářů (viz kapitola 4.5) či správy článků (viz kapitola 4.7), protože je na portálu integrován CKEditor, který ulehčuje uživateli vkládání popisů článků, seminářů a dalších prvků díky svému prostředí, které je grafické.

Uživatel tudíž nemusí znát programovací jazyk, aby byl schopen vložit do textu obrázek či zvýraznit určitý nadpis, ale vše zvládne prostřednictvím speciálního webového editoru, tzv. WYSIWYG editoru, který se svými funkcemi podobá klasickému editoru dokumentů. Jsou zde všechny standardní možnosti pro vložení nadpisů, barev pozadí či textu, odkazů, obrázků (s možností nahrání z obrázku vlastního počítače apod.). Samotné práci s tímto editorem je věnována celá kapitola v uživatelském manuálu.

```
$(function() {
    $('.show-logged-users').on('click', function() {
        window.open('pages/admin-logged-users.php?
        seminarid='+$(this).prop("id"), 'Přehled přihlášených
        uživatelů','width=800,height=auto,resizable=yes');
    });
});
```

```
<a class="show-logged-users" id="' . $item["sem_id"] . '"
title="Přehled přihlášených uživatelů"><img src="images/users.png"
alt="users" /></a>
```

Obrázek 1 – Otevření okna s přihlášenými uživateli na seminář

Výpis kódu na obrázku 1 ukazuje, jak je řešeno otevírání nového okna v případě, že uživatel klepne myší na ikonku, která má za úkol vypsat všechny uživatele, kteří jsou přihlášeni na daný seminář.

V první části tohoto výpisu kódu je vidět javascriptová funkce, která prvek s třídou *show-logged-users* obslouží událostí *click* – otevře nové okno s názvem „Přehled přihlášených uživatelů“ s určitou velikostí. Tato událost přijímá metodou *prop(„id“) id*, kde tento atribut informuje systém, pro jaký seminář se okno otevírá.

V druhé části výpisu kódu je vidět, jak vypadá odkaz, který spouští danou metodu v javascriptu. Jak je vidět, odkaz (tag *a*) má svoji třídu, která musí být stejná s názvem třídy u javascriptové metody, jinak by ke spuštění nedošlo a musí mít navíc i právě zmiňovaný atribut id, který idenfitikuje určitý seminář.

V javascriptu, konkrétně pomocí frameworku jQuery, jsou definovány i další metody, které jsou na portálu použity, zejména:

a) počítání znaků v textové oblasti (textarea),

b) nastavení i spuštění datapickeru (což je komponenta, která dokáže vypsat graficky kalendář – pro možnost výběru dat u filtru bankovních transakcí či zadávání dat při vytváření seminářů),

c) potvrzovací oznámení (opuštění semináře, smazání aktuality apod.),

d) vyjíždění a zajíždění záložek ve spodní části hlavičky,

e) zobrazení bloku s formulářem po stisku tlačítka,

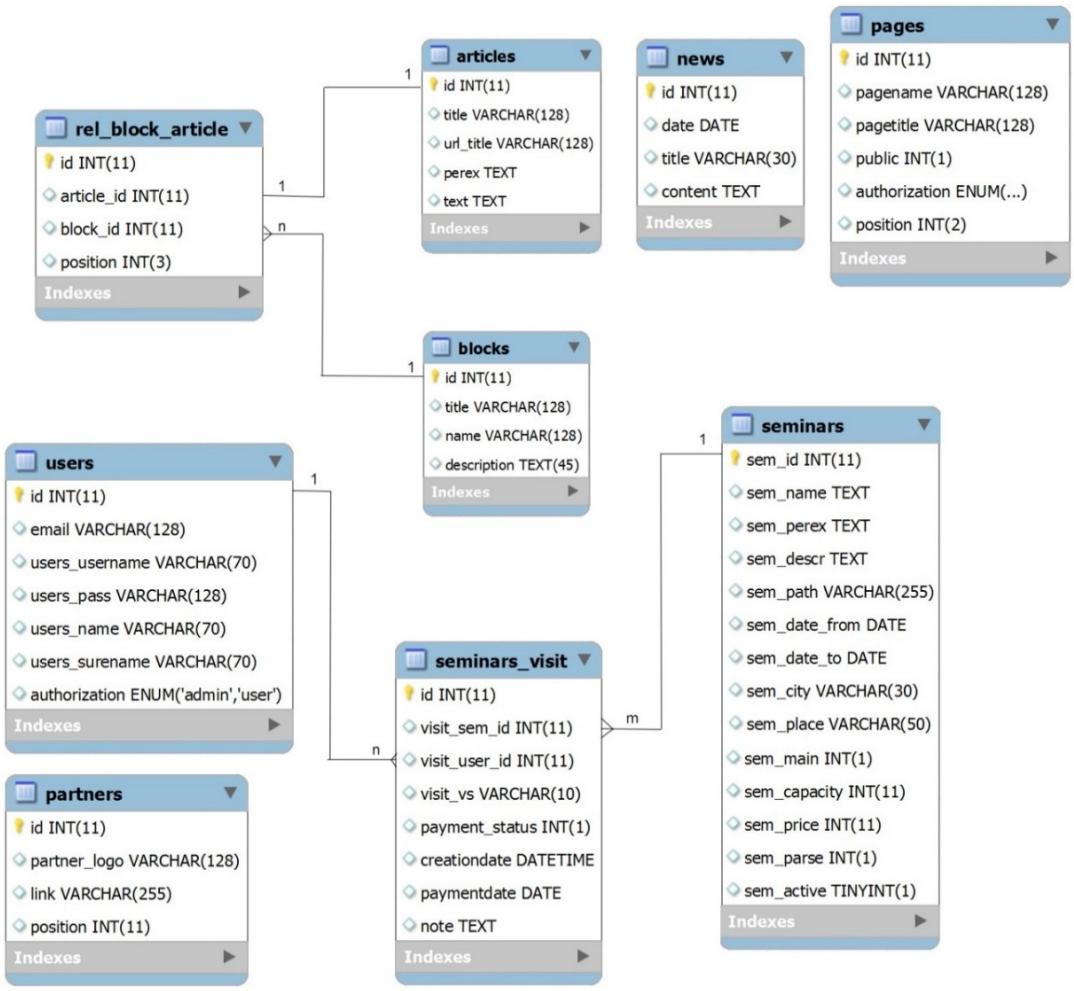
f) zmizení informační hlášky po určitém časovém úseku.

3.2 Návrh databázového modelu

Pro návrh samotné databáze je nutné nejprve zvolit vhodný databázový model. Pro tento webový portál byl zvolen relační model databáze, který je založen na tabulkách a vazbách mezi nimi. Databáze je důležitá část systému, neboť v tomto úložišti se budou uchovávat všechna potřebná data k provozu portálu.

3.2.1 ER-diagram

ER-diagram je důležitá komponenta návrhu databáze, neboť popisuje vztahy mezi jednotlivými entitami databáze, viz obrázek 2.



Obrázek 2 – ER diagram

3.2.1.1 Entity portálu

Entity portálu jsou pevně definované množiny dat, které popisují vždy konkrétní dílčí problém či množinu problému.

Articles

Entita *„Articles“* slouží na portálu k uchovávání vytvořených článků administrátorem. Články se poté zobrazují v příslušném bloku.

Obrázek 3 – Entita articles

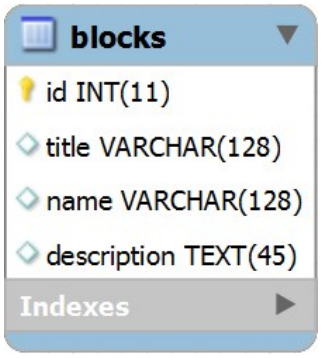
Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,
2) „*title*“ – titulek stránky, který se zobrazuje na stránce, resp. záložce prohlížeče,
3) „*url_title*“ – titulek stránky pro systém (na základě tohoto atributu ji systém identifikuje),
4) „*perex*“ – krátký text popisující obsah článku,
5) „*text*“ – samotný text článku v určité struktuře.

Blocks

Entita sloužící k definici bloků na portálu, do kterých se zobrazují vytvořené články. Mají pevně definovaný obsah, který je:

a) Home – blok „home“ je pro zobrazení stejnojmenného článku na domovské stránce,
b) Levý panel – tento blok se nachází na levé straně obsahové části,
c) Záložky – blok se záložkami je umístěn ve spodní části hlavičky.



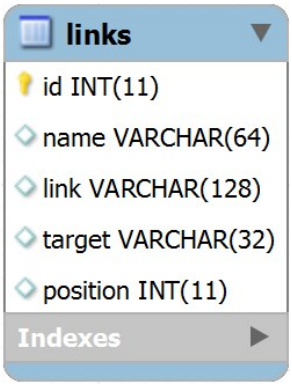
Obrázek 4 – Entita blocks

Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,
2) „*title*“ – titulek stránky, který se zobrazuje na stránce, resp. záložce prohlížeče,
3) „*name*“ – titulek stránky pro systém (na základě tohoto atributu ji systém identifikuje),
4) „*description*“ – popisem stránky pro lepší orientaci administrátora.

Links

Entita „*Links*“ slouží k uchovávání záznamů, resp. webových stránek, o kterých si administrátor myslí, že jsou důležité pro návštěvníky portálu. Záznamy se návštěvníkovi portálu zobrazí na konkrétním definovaném místě v zápatí (viz kapitola 3.1.1.6).



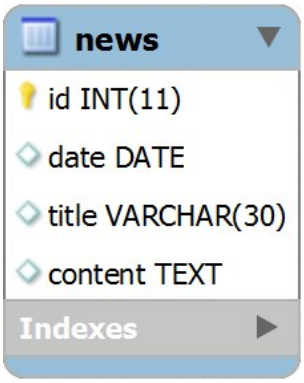
Obrázek 5 – Entita links

Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,
2) „*name*“ – název webu, který se zobrazí na samotné stránce,
3) „*link*“ – URL cesta k webové stránce (internetová adresa),
4) „*target*“ – atribut definuje, jak se otevře odkaz po označení (zda v novém okně, nebo v tomtéž apod.),
5) „*position*“ – tento atribut umožní seřadit vytvořené odkazy ve výpisu na stránce (podle číselné hodnoty).

News

Entita „*news*“ slouží pro uchovávání aktualit, které vytvoří administrátor na portálu. Umožnují uživateli (návštěvníkovi) sdělit určitou skutečnost či událost (co se děje, nově vytvořený kurz apod.)



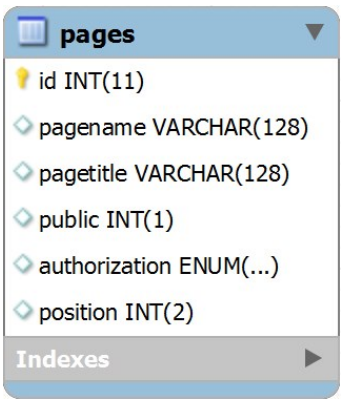
Obrázek 6 – Entita news

Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,
2) „*date*“ – atribut sloužící k uchování datumu vytvoření aktuality,
3) „*title*“ – nadpis aktuality,
4) „*content*“ – kompletní obsah aktuality.

Pages

Důležitá entita sloužící k definici všech stránek, které mohou být v rámci portálu otevřeny. Slouží i k zajištění autorizace uživatelů, kteří mohou provést určitou operaci s daty.



Obrázek 7 – Entita pages

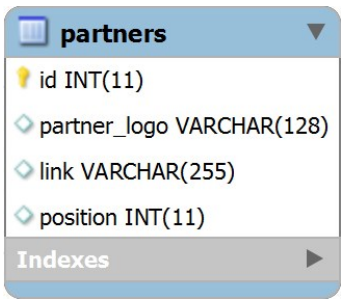
Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,

2) „*pagename*“ – titulek stránky pro systém (na základě tohoto atributu ji systém identifikuje),
3) „*pagetitle*“ – titulek stránky, který se zobrazuje na stránce, resp. záložce prohlížeče,
4) „*public*“ – atribut rozdělí stránky na veřejné a dostupné po přihlášení (1 – veřejná, 0 – po přihlášení),
5) „*authorization*“ – umožňuje určit, pro jakou uživatelskou roli je stránka dostupná (administrátor, uživatel, kdokoliv),
6) „*position*“ – tento atribut umožní seřadit vytvořené stránky podle číselné hodnoty.

Partners

Entita „*Partners*“ slouží pro ukládání údajů partnerů, se kterými administrátor spolupracuje, jak např. v rámci spolupráce na seminářích, tak na školeních či dalších akcích.



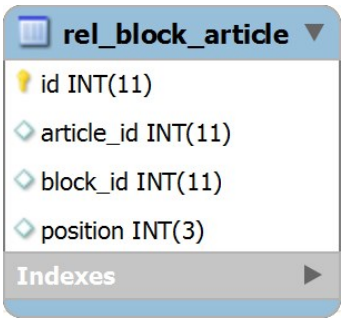
Obrázek 8 – Entita partners

Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,
2) „*partner_logo*“ – tento atribut ukládá název obrázku, který je zobrazen jako hlavní číst partnera v zápatí včetně přípony obrázku,
3) „*link*“ – atribut uchovává URL adresu webové stránky partnera,
4) „*position*“ – tento atribut umožní seřadit vytvořené odkazy ve výpisu na stránce (podle číselné hodnoty).

Rel_block_article

Tato entita slouží ke správnému zařazení článků do příslušného bloku.



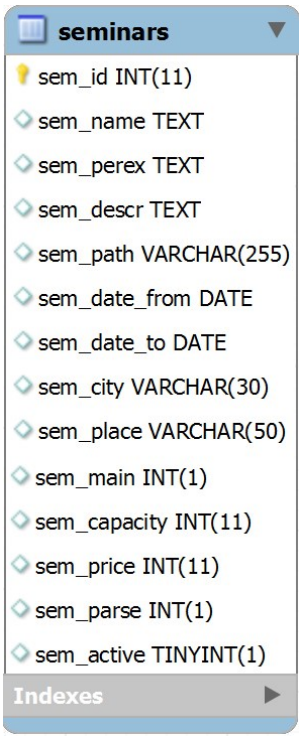
Obrázek 9 – Entita rel_block_article

Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,
2) „*article_id*“ – slouží k uchování čísla (identifikátoru) konkrétního článku,
3) „*block_id*“ – slouží k uchování čísla (identifikátoru) konkrétního bloku,
4) „*position*“ – udává pořadí článku v daném bloku.

Seminars

Entita „*Seminars*“ slouží k uchování veškerých dat, které jsou potřebné k semináři. Je to jedna z nejdůležitějších entit na portálu.



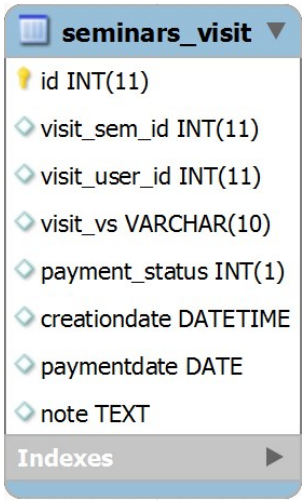
Obrázek 10 – Entita seminars

Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,
2) „*sem_name*“ – uchovává název semináře,
3) „*sem_perex*“ – krátký popis vystihující celou část obsahu semináře,
4) „*sem_descr*“ – kompletní popis semináře,
5) „*sem_path*“ – cesta k názvům souborů, které jsou přiřazeny ke kurzu (JPG soubor a PDF soubor),
6) „*sem_date_from*“ – uchovává datum začátku semináře (od kdy bude probíhat),
7) „*sem_date_to*“ – uchovává datum konce semináře (do kdy bude probíhat),
8) „*sem_city*“ – informace, kde se bude daný seminář odehrávat,
9) „*sem_place*“ – bližší specifikace umístění,
10) „*sem_capacity*“ – upřesnění kapacity semináře,
11) „*sem_price*“ – ukládá cenu za seminář,
12) „*sem_parse*“ – slouží k identifikaci, zda byl seminář vytvořen nebo importován z jiných stránek,
13) „*sem_active*“ – slouží k identifikaci, zda je seminář aktivní či nikoliv (je zobrazen pro uživatele nebo jen pro administrátora v jeho administrátorském menu).

Seminars_visit

Entita „*seminars_visit*“ slouží k uchování záznamů o uživatelích, kteří jsou přihlášení na libovolný seminář, o stavu jejich platby za něj, příp. jako historie všech navštívených kurzů



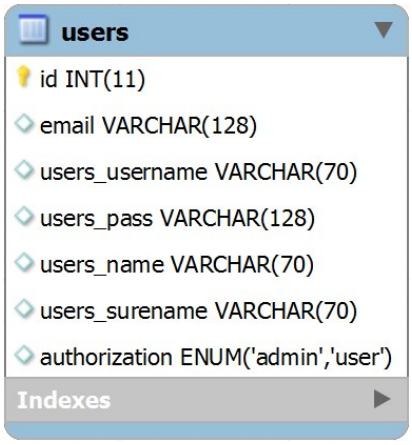
Obrázek 11 – Entita seminars_visit

Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,
2) „*visit_sem_id*“ – uchovává identifikátor semináře,
3) „*visit_user_id*“ – uchovává identifikátor uživatele,
4) „*visit_vs*“ – uchovává variabilní symbol pro daného uživatele a seminář (pro platbu),
5) „*payment_status*“ – stav platby (0 – nezaplaceno, 1 – zaplaceno, -1 – platba vrácena, -2 – nesprávná částka),
6) „*creationdate*“ – datum přihlášení uživatele (návštěvníka) na seminář,
7) „*paymentdate*“ – datum platby uživatele za seminář,
8) „*note*“ – poznámka k příslušnému uživateli u kurzu (např. zda zaplatil včas apod.).

Users

Entita „*Users*“ uchovává všechna potřebná data o klientech, kteří jsou na portál registrováni. Díky ní systém ví, jaký uživatel se na seminář přihlásil apod.



Obrázek 12 – Entita users

Popis atributů:

1) „*id*“ – slouží jako identifikátor záznamu, je zároveň i primárním klíčem v této entitě s vlastností „auto-increment“,
2) „*email*“ – slouží k uchování emailu na uživatele,
3) „*users_username*“ – uživatelské jméno uživatele,
4) „*users_pass*“ – uživatelské heslo pro uživatele,
5) „*users_name*“ – jméno uživatele,
6) „*users_surename*“ – příjmení uživatele,
7) „*authorization*“ – oprávnění uživatele (administrátor nebo uživatel).

3.2.2 DF diagram

DF diagram je silný modelovací nástroj, který je důležitý z hlediska datových a řídících toků systému. Popisuje, jak se systém vnitřně chová a jaké jsou vazby mezi úložišti dat či funkcemi.

DF diagramy obsahují čtyři základní stavební prvky:

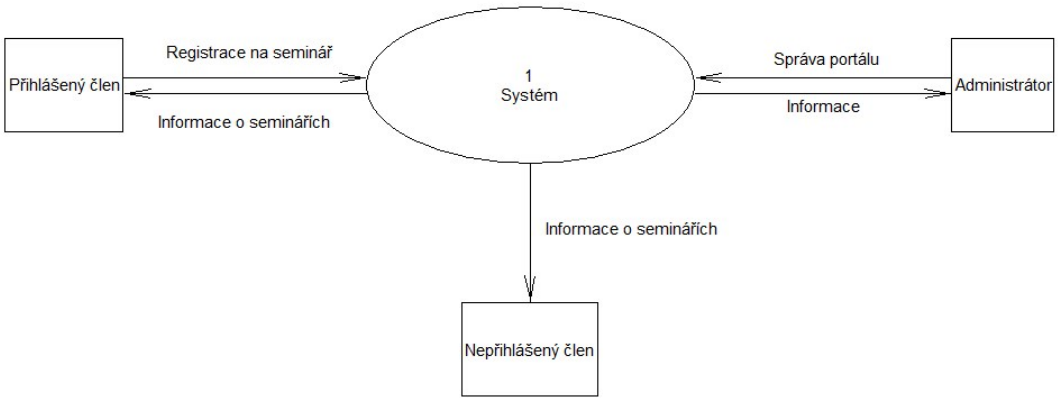
1) Terminátor – jsou to externí entity, které komunikují se systémem, ale stojí vně tohoto systému. Může to být např. organizace, skupina lidí, či jiný systém. Značí se obdélníkem, ve kterém je umístěn název terminátora.
2) Tok – toky jsou samostatné proudy dat, které po systému proudí různými směry – směrem k terminátorům, či úložištím dat. Značí se čárou s šipkou.

3) Úložiště – slouží pro uložení dat, která se přenáší pomocí toků. Může jím být např. databáze. Značí se dvěma vodorovnými čárami, kde uprostřed těchto čar je název úložiště.

4) Proces – samostatné funkce, které mění vstup na výstup. Značí se bublinou s názvem procesu a v systému provádí určitou operaci.

3.2.2.1 Kontextový diagram

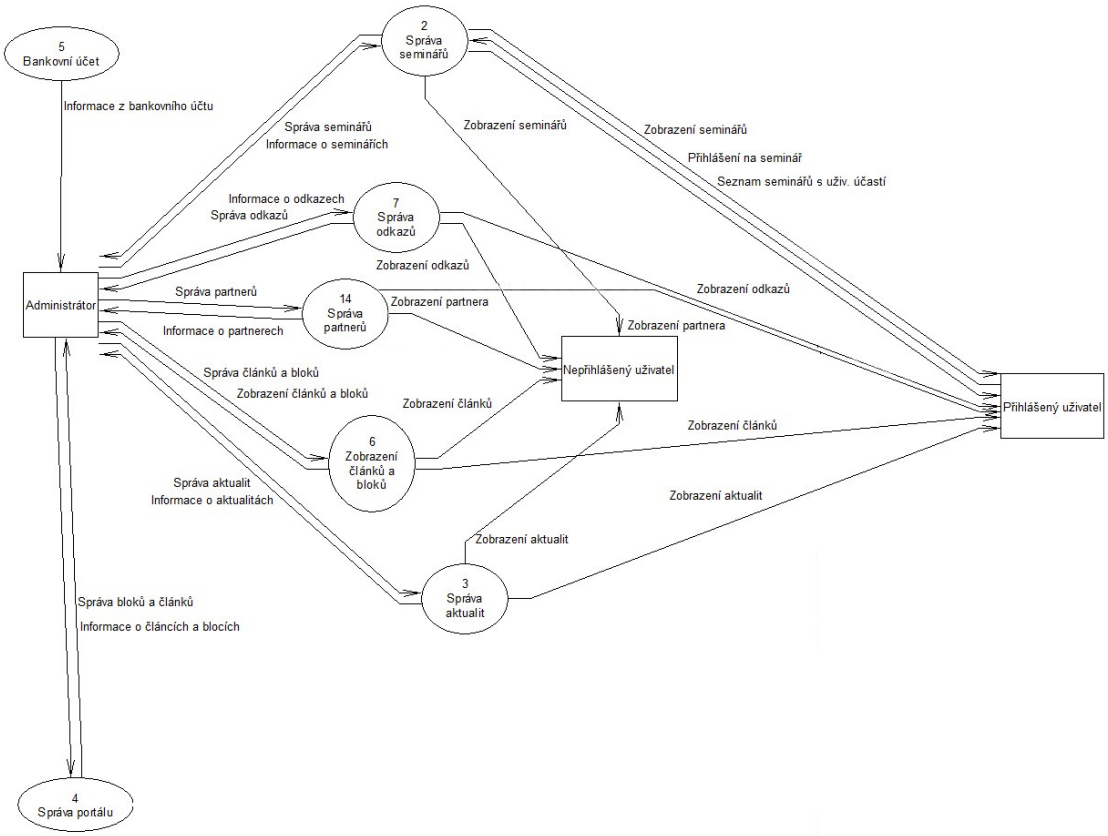
Kontextový diagram je nejvyšší v hierarchii DF diagramů. Obsahuje hlavní proces, který značí systém a dále terminátory, které se systémem komunikují, jak je vidět na obrázku 13.



Obrázek 13 – Kontextový diagram

3.2.2.2 DF diagram – 0. úroveň

Tato úroveň diagramu popisuje detaily systému, konkrétně jaké funkce jsou v systému obsaženy a jací terminátoři s nimi pracují. V této úrovni nejsou popsány detaily určitých funkcí, neboť jsou popsány o úroveň níže (tj. v 1. úrovni). Diagram 0. úrovně je zobrazen na obrázku 14.

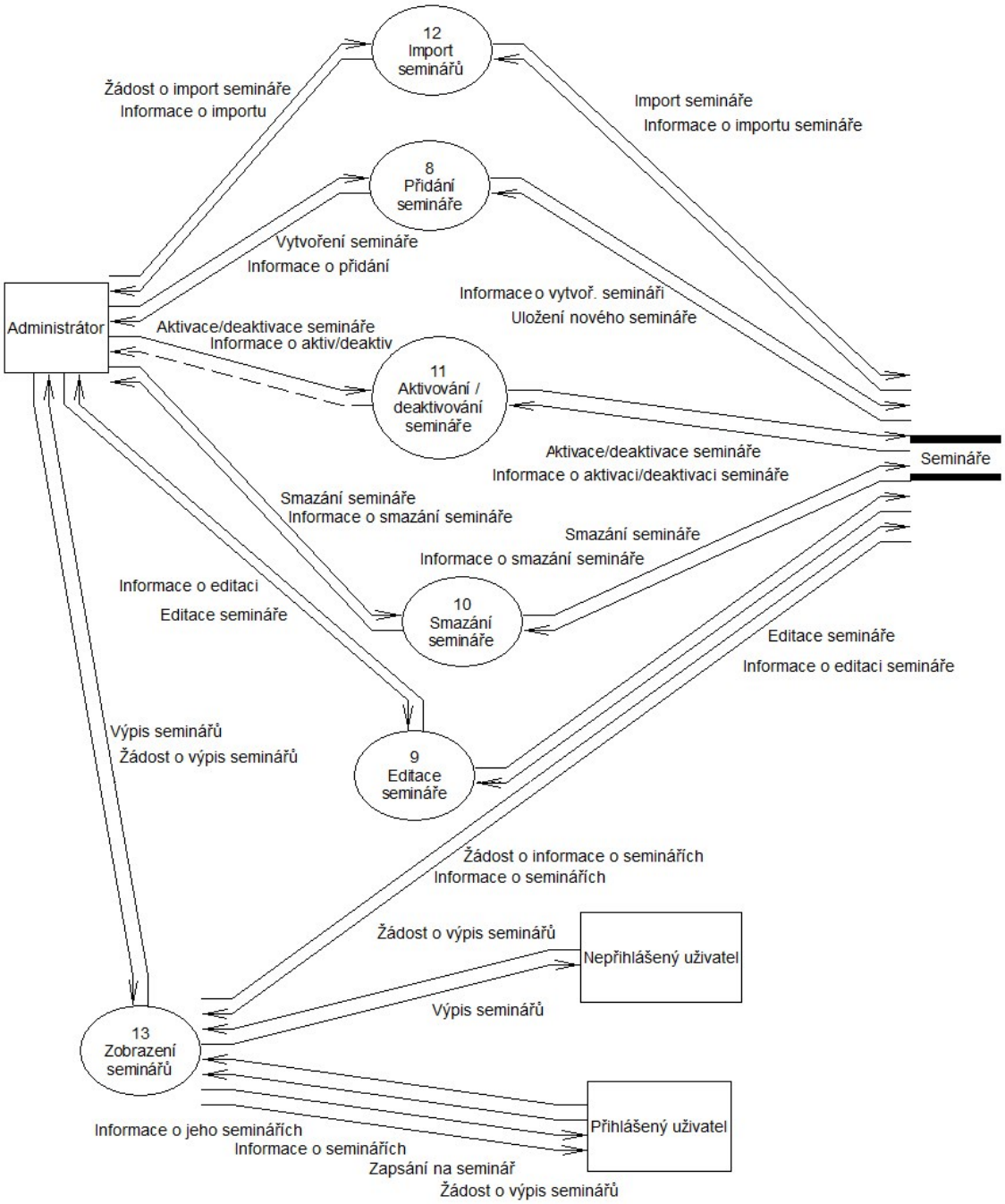


Obrázek 14 – 0. úroveň DF diagramu

3.2.2.3 DF diagram – 1. úroveň

Na obrázku 15 je vidět, že tento diagram popisuje funkci, která zajištuje na portálu správu seminářů. Je vidět, jak pracuje správa seminářů, jaké funkce obsahuje a kdo s nimi pracuje. Administrátor je schopen seminář vytvořit, editovat, smazat či aktivovat/deaktivovat a v neposlední řadě semináře importovat.

Naopak uživatel s právy nižšími, tj. pouze přihlášený uživatel, může pouze číst (pouze zobrazit seminář), resp. systém mu umožní se na něj přihlásit. Uživatel, který se nepřihlásil, má právo pouze prohlížet vytvořené semináře bez možnosti se na seminář přihlásit. V této úrovni se vyskytuje i datové úložiště, do kterého se všechny změny od administrátora ukládají.



Obrázek 15 – 1. úroveň DF diagramu – správa seminářů

4 Realizace funkcí

Samotná realizace funkcí portálu je nejdůležitější část práce, protože v tomto bloku se vytváří kompletní funkční řešení. Řeší se všechny potřebné funkce, které je třeba vytvořit, aby systém fungoval a pracoval tak, jak má na základě návrhu řešení, viz kapitola 3.

4.1 Architektura aplikace

Architektura aplikace vystihuje činnost z hlediska rozdělení funkcí do jakýchsi sekcí, které disponují určitými funkcemi, jež spolu úzce souvisí. Základem architektury jsou vždy třídy, na základě kterých vznikají objekty. Třídu lze chápat jako jakýsi vzor či definici funkcí, které bude mít daný datový objekt odvozený od této třídy (instance třídy).

4.1.1 Třídy portálu

Hlavními sekcemi tohoto portálu jsou následující třídy:

1) *class.database*,
2) *class.functions*,
3) *class.imageresize*,
4) *class.page*,
5) *class.record*.

Těchto pět hlavních tříd, potažmo objektů, zajišťuje správnou činnost portálu a starají se o chod systému. Portál obsahuje ještě další třídy, které nemají takový význam jako tyto hlavní, protože mají na starost jeden konkrétní problém – např. správa seminářů či uživatelů. Při absenci těchto menších tříd dojde na portálu k chybě pouze u dané záložky (např. uživatelé), avšak další sekce (semináře, články apod.) budou fungovat tak, jak mají. Naopak při absenci hlavních tříd nebude funkční celý systém bez ohledu na to, s čím je třeba pracovat.

Menší třídy přejímají metody z těchto hlavních tříd (dědí) a jsou rozšířeny o další specifické metody, které slouží k práci s konkrétní částí portálu.

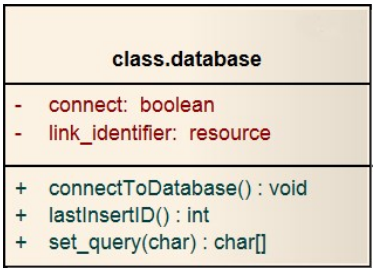
4.1.1.1 Třída „class.database“

Třída „*class.database*“ slouží k definici objektu, který pracuje s databází. Objekt zajišťuje komunikaci s úložištěm dat a obsahuje metody, které umožňují provádět libovolné SQL dotazy.

Nejdůležitější metoda třídy je navázání spojení s databází. K tomu slouží metoda *connectToDatabase*. Další metoda, kterou obsahuje tato třída, vrací identifikátor posledního vloženého záznamu (*lastInsertID*).

Poslední metodou, kterou má tato třída, je metoda *set_query*, která zajišťuje vykonání samotných SQL dotazů a vrácení výsledků, které se dále zpracovávají.

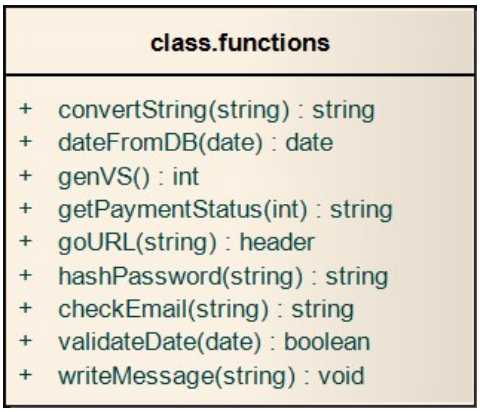
Celé schéma tohoto objektu je zobrazeno na obrázku 16.



Obrázek 16 – Třída pro práci s databází

4.1.1.2 Třída „class.functions“

Tato třída je pro správný chod portálu důležitá, neboť poskytuje metody, které pracují s daty (převádějí vzájemně formáty pro výpis a pro databázi), realizují zahashování hesla, poskytují kontroly správnosti emailové adresy či URL a v neposlední řadě umožní přesměrování na konkrétní jinou stránku.



Obrázek 17 – Třída pro funkce portálu

Nejzajímavější metoda této třídy má název *genVs*. Jak už je z názvu patrno, metoda generuje variabilní symbol pro uživatele, který se přihlásil na konkrétní seminář.

Variabilní symbol je taková posloupnost čísel, která se nesmí opakovat, jinak by mohlo dojít k nekonzistentnosti platby. Jak je vidět na výpisu kódu na obrázku 18, variabilní symbol je kombinací aktuálního data a času v době, kdy se uživatel na seminář přihlásil a navíc ještě součtem s náhodně vygenerovaným číslem. Tato kombinace zaručuje, že pokud nenastane situace, kdy se v jednu chvíli přihlásí na seminář více jak 99 999 lidí, variabilní symbol bude vždy unikátní.

```
function genVS() {
    $part1 = date("y") + date("m"); // součet posledního dvojčíslí
    roku a měsíce
    $part2 = date("d"); //den
    $part3 = date("His") + rand (00000, 99999); //součet hodina,
    minuta, sekunda a náhodné 5místné číslo
    return $part1 . $part2 . $part3;
}
```

Obrázek 18 – Metoda pro generování variabilního symbolu

Další důležitou metodou v této třídě je metoda pro hashování hesla, které se ukládá do databáze. Tato metoda se také využívá při přihlašování uživatele. Funkce se nazývá *hashPassword*, má pouze jeden vstupní parametr, kterým je řetězec znaků.

Funkce tento řetězec převede pomocí speciálního hashovacího algoritmu SHA-512 na hashovou formu a výslednou posloupnost znaků vrátí. Vzhledem k tomu, že se jedná o jednocestný algoritmus, z vytvořeného hashe se nedá získat zpět původní heslo.

Jak celý algoritmus funguje je zobrazeno ve výpisu kódu na obrázku 19.

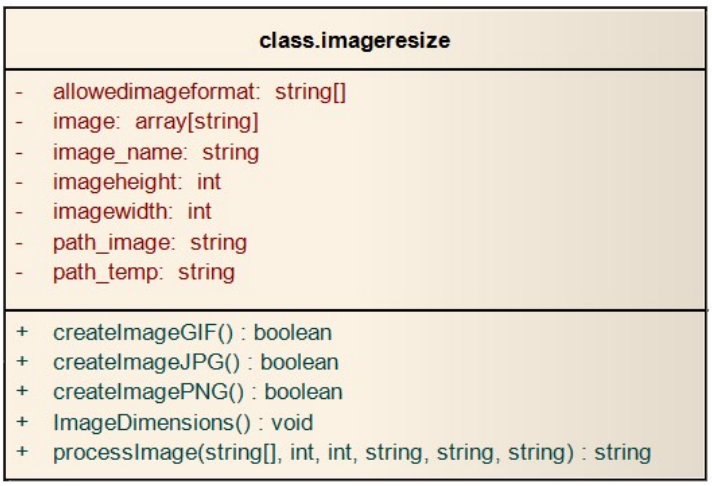
```
function hashPassword2($password) {
    $passHash=hash("sha512",$password);
    return $passHash;
}
```

Obrázek 19 – Hashování hesla

4.1.1.3 Třída „class.imageresize“

Třída „*class.imageresize*“ je základní třída, které slouží pro práci s obrázky. Umožňuje nahrávat obrázky na server do konkrétní složky, daný obrázek zmenšit, resp. upravit mu rozměry či vytvořit obrázek jako duplikát v jiném rozměru pro zobrazení náhledu.

Celá třída včetně všech metod je zobrazena na obrázku 20. Třída obsahuje několik atributů, které slouží k uchovávání dat o obrázku, názvu, rozměrech, ukládání dočasného souboru při nahrávání na server a v neposlední řadě i výčet povolených formátů.



Obrázek 20 – Třída pracující s obrázky

Zajímavější část této třídy jsou metody, které třída realizuje.

Metoda *processImage* umožňuje obrázek nahrát na server, přičemž musí být dodrženy základní podmínky pro velikost obrázků (musí být dodržena minimální velikost) a správná přípona daného obrázku.

```
// Nahrátí obrázku na server do dočasné složky
    if(!move_uploaded_file($this->loadedimage["tmp_name"], $this->
    path_temp.$this->loadedimage["name"])) {
        return "<p>Nepodařilo se nahrát obrázek na server.</p>";
    } else {
        // Provede výpočet nových rozměrů
        $this->ImageDimensions(); //volá interní funkci
        // Vybere funkci, kterou použije na vytvoření obrázku
        switch($this->loadedimage["imagesize"]["mime"]) {
            case "image/jpeg":
                if($this->imageformat == 'thesame') $this->
                imageformat = 'jpg';
```

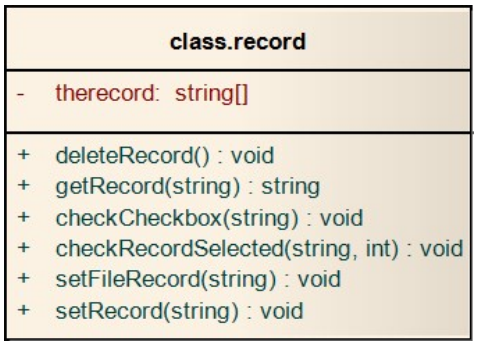
Obrázek 21 – Část metody pro nahrání obrázku

4.1.1.4 Třída „class.record“

Třída „*class.record*“ slouží pro práci s editovanou položkou, přičemž je jedno, zda se jedná o aktualitu, nebo např. seminář. Při libovolné operaci, se data z formuláře uloží do tohoto objektu pro další možnost zpracování.

Třída obsahuje hlavní atribut, což je pole, které se plní podle potřeby.

Metody této třídy jsou určeny právě ke správě daného pole, vkládání dat, pro snadnější orientaci, pro kontrolu zaškrtávacích tlačítek. Je zde i metoda na vyčistění pole (smazání obsahu) tohoto pole.



Obrázek 22 – Třída pro práci s daty z formulářů

Zajímavou metodou této třídy je *getRecord*, která zajišťuje korektní vrácení dané hodnoty pod určitým klíčem. To znamená, že vrátí data, která obsahuje pomocné pole, pokud se v tomto poli vyskytuje klíč, který předáváme do funkce.

```
function getRecord($key) {
     $thereturn = null;  // nastavení návratové hodnoty
     if(is_array($this->therecord)) { //pokud je pole, je to editace
          if(array_key_exists($key, $this->therecord)) //kontroluje
          v poli existenci daného klíče
          $thereturn = $this->therecord[$key]; //pokud existuje klíč,
          vrátí jeho hodnotu
     }
     return $thereturn;
}
```

Obrázek 23 – Získání dat z pole

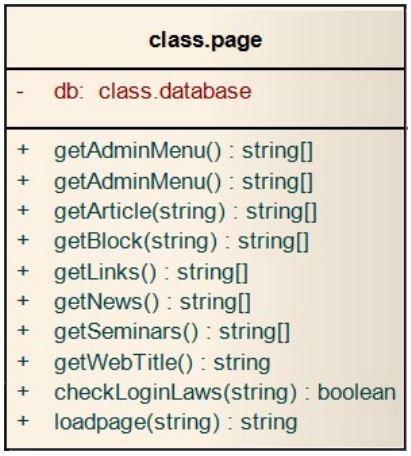
4.1.1.5 Třída „class.page“

Neméně důležitá je třída „*class.page*“, které slouží pro veškerou práci se samotnými stránkami na portálu. Díky této třídě systém dokáže zjistit, zda přihlášený

uživatel má právo přistoupit na konkrétní stránku (např. stránku určenou pouze pro administrátora).

Třída disponuje i metodami, které zjistí data, jež obsahuje administrátorské menu, jaký je název aktuální stránky či data se všemi dostupnými semináři.

Tato třída, jak je vidět z logické skladby funkcí, pracuje s databází, je tedy nutné, aby obsahovala alespoň jeden atribut, kterým bude samotné spojení s tímto úložištěm dat, konkrétně celou instanci třídy pro práci s databází (viz kapitola 4.1.1.1).



Obrázek 24 – Třída pro zobrazování dat

Důležitá je i samotná definice třídy, neboť tato třída dědí z třídy „*class.functions*“ (viz kapitola 4.1.1.2), to tedy znamená, že třída má všechny metody třídy s funkcemi a navíc je rozšířena ještě svými metodami, které jsou vykresleny na obrázku 24.

Metody jsou víceméně podobné a slouží převážně k zajišťování dat z databáze, zejména data o kurzech, článcích, obsahu v daném bloku a seminářů.

Výjimku tvoří tři metody, které nejsou implementovány pouze za účelem získání konkrétních dat.

Metoda *getWebTitle*, slouží k získání informace o titulku stánky, na kterou uživatel přešel. Tento nadpis stránky se zobrazuje v hlavičku celého portálu, konkrétně v tagu <*title*>.

Druhou zajímavější metodou je metoda *loadpage*, která zajištuje správné zobrazení stránky uživatelům podle jejich práv. Funkce zjistí, zda se jedná o veřejnou stránku, pokud ano, zobrazí ji uživateli, pokud ne, zjistí, zda přihlášený uživatel má práva, která umožňují otevřít tuto stránku.

Třetí a poslední metoda se nazývá *checkLoginLaws*, kterou volá předešlá funkce pro získání uživatelských práv pro zajištění správného otevření autorizovanému uživateli.

Daná funkce zjistí, zda je uživatel přihlášen, pokud ano, musí mít pro stránku, která je do této metody předávána, uživatelská práva. Celý průběh zjištění uživatelských práv je vidět ve výpisu kódu na obrázku 25.

```
function checkLoginLaws($page_authorization) {
     if(isset($_SESSION["userloggedin"])) { //zjištění přihlášení
          if($page_authorization == 'everybody') //pokud je pro každého
               return true;
          if($_SESSION["userloggedin"]["authorization"] ==
          $page_authorization)
               return true;
     }
     return false;
}
```

Obrázek 25 – Zjištění uživatelských práv

4.1.2 Jádro systému

Samotné objekty nestačí pro správnou činnost systému. K té je třeba vytvořit instance těchto tříd, to znamená vytvořit konkrétní datové objekty, které jsou z těchto tříd odvozeny a které mohou být využity pro práci uvnitř portálu.

O tuto činnost se stará soubor *core.php*, v překladu jádro, které zajišťuje správnou inicializaci celého systému včetně zjištění, zda některá z důležitých tříd není dostupná.

Samotné spuštění jádra se provádí ihned po otevření portálu uživatelem, což zaručí, že při výskytu chyby se uživateli zobrazí chybová hláška.

Tento soubor obsahuje pouze již zmíněné definice instancí tříd a jednu událost, která je důležitá zejména pro administrátora a klienty, kteří se chtějí přihlásit na seminář. Tou funkcí je **přihlášení.**

```
if(is_file('inc/settings.php')) include_once 'inc/settings.php';
if(is_file('inc/class.functions.php'))include_once
'inc/class.functions.php';
if(is_file('inc/class.database.php')) include_once
'inc/class.database.php';
if(is_file('inc/class.record.php'))include_once
'inc/class.record.php';
if(is_file('inc/class.page.php'))include_once 'inc/class.page.php';

$webfunc = new tFunctions();
$db = new tDatabase();
$page = new tPage($db);
```

Obrázek 26 – Systémový soubor core.php

Neméně důležitým souborem je soubor *settings.php*, kde je uloženo veškeré nastavení týkající se portálu. Jsou zde uloženy přístupy k databázi, bankovnímu účtu, rozměry obrázků, počet aktualit, které se na stránce vykreslují či cesta, pro ukládání obrázků atd.

```
define('SEMINARS_IMG_WIDTH',140); // šířka obrázků u seminářů
define('SEMINARS_IMG_HEIGHT',140); //výška obrázků u seminářů
define('SEMINARS_PATH', 'seminars/'); //cesta pro obrázky a PDF
soubory
define('BANK_ACCOUNT', '0123456789/0123'); // číslo bankovního učtu,
který bude uveden v emailu pro přihlášeného uživatele
```

Obrázek 27 – Struktura souboru settings.php

4.2 Autentizace

Autentizace je proces, kdy se ověřuje identita uživatele, přičemž samotných druhů autentizací je několik. Pro portál byla zvolena autentizace formou hesla a uživatelského jména. Ověření identity má na starost třída *UserLogin,* která má pouze jednu metodu pro toto ověření a jeden atribut, resp. instanci třídy pracující s databází, neboť při samotném přihlášení je nezbytné provést kontrolu dat právě z úložiště.

4.3 Autorizace

Autorizace uživatele je proces, který ověří, zda uživatel má právo na zobrazení konkrétní stránky či provedení určité funkce. Tento proces na portálu zajišťuje hlavní třída pracující se stránkami, která je popsána výše (viz kapitola 4.1.1.5).

4.4 Osobní menu

Toto menu se nachází v administrátorském menu a je viditelné každému přihlášenému uživateli. Jsou to první dvě políčka, která umožní uživateli nahlédnout do svých údajů, které uvedl při registraci (např. při zapomenutí emailu, na který se registroval), či ke zjištění na jaké semináře je uživatel zapsán nebo jaké navštívil.

4.4.1 Můj profil

První položka v menu slouží uživateli k nahlédnutí do vlastních údajů, kde se nachází i možnost změnit heslo. Změna hesla je důležitá v případě, že uživatel heslo zapomněl a nechal si vygenerovat nové, na základě kterého se do systému dostal, nicméně si toto heslo potřebuje změnit, aby si ho lépe pamatoval. Může se také stát, že se heslo nedopatřením dozví cizí osoba. V těchto a podobných případech je důležité si heslo změnit.

Po stisku tlačítka „Změna hesla“ se uživateli objeví formulář, kam musí zadat stávající staré heslo a nové heslo. O samotnou změnu hesla se stará metoda *editPass* (část funkce je zobrazena ve výpisu kódu na obrázku 28), která jako jediný vstupní atribut potřebuje číslo – id daného uživatele. Na základě tohoto id se v databázi zjistí, zda staré heslo souhlasí a jsou identická i políčka nového hesla a hesla pro kontrolu. Pokud je vše splněno, do databáze je uložena nová hodnota hesla v hashové podobě, což slouží jako ochrana proti přečtení z databáze.

```
function editPass($userid) {
    $q = "SELECT users_pass FROM users WHERE id = '$userid'";
    // SQL dotaz, zda existuje dany uživatel s konkrétním id
    $r = $this->db->set_query($q);
    if(count($r[0]) > 0 && $r !== false) { //pokud existuje
        if($this->func->hashPassword2($this->therecord["old_pass"])
        == $r[0]["users_pass"]) { //porovnání starého hesla a hesla
        z formuláře
```

Obrázek 28 – Změna hesla

4.4.2 Moje semináře

Položka v menu „Moje semináře“ slouží k výpisu seminářů, na které je přihlášen či na které byl přihlášen daný uživatel. Je zde výpis všech dostupných seminářů včetně většiny atributů, zejména název, místo semináře, město, cena semináře a v neposlední řadě stav platby – zda uživatel zaplatil či nikoliv nebo možnost opustit tento seminář, pokud ještě nebyl zaplacen.

4.5 Semináře

4.5.1 Semináře vytvořené v rámci portálu

Hlavní částí celého portálu je možnost spravovat semináře, tzn. semináře přidávat, editovat či mazat. Administrátor musí mít přehled, kdo je na jaký seminář zapsán nebo jaký je stav plateb od přihlášených uživatelů.

Kompletní správu má na starosti třída *tSeminars*, která dědí od hlavní třídy *class.record* (viz kapitola 4.1.1.4) a navíc obsahuje všechny metody pro obsluhu operací se semináři, tzn. metody pro přidání, editace, aktivování/deaktivování semináře atd.

Základem pro správnou činnost je vytvoření instance této třídy.

```
$seminars = new tSeminars($db,$webfunc);
```

Obrázek 29 – Instance třídy tSeminars

Jak je vidět ve výpisu kódu na obrázku 29, objekt vytvořený z této třídy má dva parametry. Prvním parametrem je parametr *db*, což je objekt, na základě kterého můžeme využívat všechny dostupné operace a metody zajišťující práci s databází, třídou *class.database* (viz kapitola 4.1.1.2).

Druhým parametrem je *webfunc*, což je opět objekt, tentokrát poskytující přístup k metodám třídy *class.function* (viz kapitola 4.1.1.1).

```
function editSeminar($seminarid) {
    $q = "SELECT sem_id, sem_name, sem_perex, sem_descr, sem_path,
    sem_date_from, sem_date_to, sem_city, sem_place, sem_main,
    sem_capacity, sem_price, sem_parse, sem_active FROM seminars WHERE
    sem_id = '" . intval($seminarid) . "'";
    $qr = $this->db->set_query($q); // provedení SQL dotazu
    if(count($qr[0]) > 0 && $qr !== false) { //pokud byl proveden
        $this->setRecord($qr[0]);
```

```
            $this->therecord["sem_date_from"]    = $this->func->
            dateFromDB($this->therecord["sem_date_from"]);
            $this->therecord["sem_date_to"]      = $this->func->
            dateFromDB($this->therecord["sem_date_to"]); //uprava datumu
        } else {
            return false;
        }
}
```

Obrázek 30 – Metoda pro editaci semináře

Jak je vidět ve výpisu kódu na obrázku 30, metoda obsahuje jediný parametr, což je identifikátor konkrétního semináře. Je proveden SQL dotaz na databázi, který vrátí veškeré informace o semináři a pomocí metody *setRecord* je uloží do pomocné proměnné. Za povšimnutí stojí funkce *intval*, která se nachází v SQL dotazu a převede řetězec na číslo. Dále funkce pouze zamění formát data z podoby „2014-05-05“ do čitelnější podoby „05.05.2014“.

Kromě samotných metod, které zajišťují správu seminářů, jsou zde implementovány i metody, které zjistí, zda došlo na portálu (konkrétně v rámci seminářů) k odeslání nějakého z formulářů. Funkce tedy zjistí, zda došlo např. k přidání nového semináře, editaci stávajícího, nebo např. kopírování. Na základě této informace zavolá příslušnou metodu pro vykonání operace.

4.5.2 Import seminářů

Import seminářů je velice specifická funkce, která slouží k rozšíření stávajících funkcí systému.

Funkce zjistí, zda na konkrétní stránce existuje seminář či semináře. V kladném případě všechny existující data stáhne a vloží na portál.

Tuto činnost má na starost třída *tImportSeminars*, která přejímá metody z třídy *class.function* s přidaným atributem, což je objekt pracující s databází (budeme semináře přímo vkládat do databáze).

Třída má metody určeny zejména ke stažení dat a ukládání do databáze, zejména metoda *importNewSeminars*, což je hlavní metoda, který realizuje všechny potřebné funkce a pracuje s daty. Tato metoda volá další metody jako např. *getCalendarActions* – metoda, která se stará o samotné parsování dat z jiné stránky nebo funkce

downloadData, které data stáhne. Této metodě se předá pouze url a funkce vrátí v poli výsledek operace.

Metoda *getCalendarActions* tyto data naparsuje (rozdělí data podle skupin apod.) a uloží do pole pro možné zpracování hlavní třídou *importNewSeminars*, která data z pole postupně ukládá do databáze, konkrétně do tabulky se semináři.

```
function importNewSeminars($returnhtml) {
     // Nastavení proměnné pro výstup z funkce getCalendarActions()
     $this->return_html = $returnhtml;
     $actions = $this->getCalendarActions();
     if(!is_array($actions)) { //pokud výsledek není pole
          return "Pravděpodobně došlo k chybě. <br />Opakujte akci
          později se obraťte na správce webu";
     } else {
          $totalcount = count($actions); //počet výsledků
⋮
```

Obrázek 31 – Ukázka importu seminářů

4.6 Aktuality

Sekce aktuality slouží k rychlému přístupu k informacím návštěvníků, co se na webu za poslední dobu událo. Aktuality se vypisují v pravém bloku obsahu (viz kapitola 3.1.1.5), což zajišťuje, že uživatel ihned po otevření portálu jej uvidí.

Správu aktualit obstarává třída *tAdminNews*, která disponuje metodami umožňujícími aktualitu přidat, smazat či editovat.

Instance této třídy obsahuje stejné parametry, jako tomu bylo u seminářů výše (viz kapitola 4.4) a to objekty pro práci s databází a s funkcemi. Opět přejímá metody z třídy *class.record* (viz kapitola 4.1.1.4).

```
function deleteAdminNews($newsid) {
     $q = "DELETE FROM news
          WHERE id = '" . intval($newsid) . "'";
     if($this->db->set_query($q)) //provede samotné smazání z databáze
          return "Aktualita byla úspěšně smazána"; //návratová hláška
     else
```

```
            return "Aktualitu se <strong>nepodařilo</strong>smazat.
            <br />Opakujte akci nebo se obraťte na správce webu";
}
```

Obrázek 32 – Ukázka mazání aktualit

Metoda obsahuje pouze jeden parametr, což je identifikátor dané aktuality. Na základě tohoto identifikátoru se provede dotaz, který příkazem *delete* odstraní danou aktualitu z databáze. Administrátor je o stavu operace informován výpisem o úspěchu/neúspěchu.

Tento objekt zajišťuje i zjištění, zda byl odeslán určitý formulář. Na základě této operace volá právě konkrétní metodu při správě aktualit.

```
if(isset($_GET["deleterecord"]))
     $returnmessage = $news->deleteAdminNews($_GET["deleterecord"]);
```

Obrázek 33 – Operace smazání aktuality

Jak je vidět ve výpisu kódu na obrázku 33**,** systém kontroluje existenci proměnné *deleterecord*. Pokud je tato proměnná nastavena, zavolá se metoda *deleteAdminNews* (viz výpis kódu na obrázku 32) s právě jedním parametrem, kterým je identifikátor dané aktuality). Daná operace se provede a výsledek (v tomto případě se z funkce vrátí řetězec znaků) je předán do proměnné *returnmessage*. Na každé stránce kam je uživatel přesměrován se kontroluje, zda tato proměnná obsahuje nějaká data, pokud ano, vypíšou se administrátorovi pomocí metody *writeMessage*, která je implementována ve třídě *class.functions* (viz kapitola 4.1.1.2).

4.7 Články

Vzhledem k tomu, že je celý portál postaven ve stylu redakčního systému, umožňuje vytvářet články, které se berou jako jakýsi text, který se v určitém místě zobrazí. Je jedno, kam je článek určen, má vždy stejnou strukturu, jako název apod. a až sám administrátor ho použije v konkrétním bloku pro zobrazení – viz kapitola 4.8.

Správu článků má na starost třída *tArticles*, která dědí všechny metody ze třídy *class.record.* Při vytváření instance této třídy jsou použity opět dva parametry a to objekt pro práci s databází a s funkcemi. Třída poskytuje administrátorovi metody pro vytváření článků, editaci či smazání. Implementována je i metoda, která vrací seznam všech existujících článků z databáze.

```
function saveArticle() {
    $url_title = $this->func->convertString($this->
    therecord["title"]); //převede název na korektní url
    if(array_key_exists('id', $this->therecord)) { //pokud existuje,
    je to editace
        $querystatement = "UPDATE articles SET title = '" .
        mysql_real_escape_string($this->therecord["title"]).
        "',url_title   = '" . mysql_real_escape_string($url_title) .
        "', perex       = '" . mysql_real_escape_string($this->
        therecord["perex"]) . "', text        = '" .
        mysql_real_escape_string($this->therecord["text"]) . "'WHERE
        id    = '" . intval($this->therecord["id"]) . "'";
    } else { //jedna o vložení nového článku
        $querystatement = "INSERT INTO articles (title,url_title,
        perex,text) VALUES('" . mysql_real_escape_string($this-
        >therecord["title"]) . "', '" .
        mysql_real_escape_string($url_title) . "', '" .
        mysql_real_escape_string($this->therecord["perex"]) ."', " .
        mysql_real_escape_string($this->therecord["text"]) . "')";
    }
    if($this->db->set_query($querystatement)) { //provedení SQL
        $this->deleteRecord ();
        return "Článek byl úspěšně uložen.";
    } else { return "Článek se <strong>nepodařilo</strong> uložit.";}}
```

Obrázek 34 – Uložení článku

Výpis kódu na obrázku 34 ukazuje, jak vypadá metoda, která ukládá článek do databáze. V první části převede název článku na korektní URL (bez diakritiky, bílých znaků apod.) pomocí metody *convertString*, dále zjistí, zda se jedná o editaci stávajícího článku nebo vytvoření nového. Sestaví dotaz, který následně provede a o jeho úspěšnosti informuje administrátora pomocí hlášky.

Článek nelze z portálu smazat, dokud má přidělen blok. Je tedy nezbytné v první řadě vymazat článek z bloků a poté smazat samotný článek ve správě článků.

Třída samozřejmě kontroluje, zda došlo k odeslání formuláře, potažmo, jaká událost toto odeslání vyvolala, zda administrátor stiskl vytvoření nového článku nebo smazání apod. Na základě této informace zavolá příslušnou metodu, která vše zpracuje.

4.8 Bloky

Bloky jsou úzce spojeny se samotnými články, protože článek, který administrátor vytvoří, je následně zobrazen v této sekci a je nutné mu přiřadit konkrétní zobrazovací blok.

Na portálu jsou implementovány tři hlavní bloky a to:

a) Levý panel – nachází se na levé straně v obsahové části.

b) Záložky – nachází se na spodní části hlavičky s obrázkem.

c) Home – samotný blok home spolu s článkem home tvoří obsah úvodní stránky.

Pro možnost správy bloku je administrátorovi poskytnuto grafické rozhraní, kde může vybrat konkrétní článek a jemu přiřadit patřičný blok. Je nutné dodržet podmínky vkládání, např. to že musí být počet článků v záložkách roven hodnotě 4.

Třída poskytuje metody pro uložení článku do bloku, editace článku v daném bloku či jeho smazání z bloku. Jsou zde implementovány i metody, které vrací všechny články v konkrétním bloku či metoda, který vrací všechny dostupné články pro případ, že by administrátor vytvořil článek a chtěl by ho někde zobrazit – přidělit mu určitý blok.

```
if(isset($_POST["save_block"])) { //byl odeslán formulář
     $blocks->setRecord($_POST); //uloží obsah formuláře do pole
     $returnmessage = '';
     $is_ok = true;
     if(!is_numeric($_POST["position"])) { // pokud není číslo - chyba
          $returnmessage.= '<p accesskey="position">Nebyla vyplněna
          pozice odkazu (nutno zadávat číslo).</p>';
          $is_ok = false;
     }
     if($is_ok) {
          $returnmessage = $blocks->saveBlock(); //volá metodu pro
          uložení článku do bloku
     } else { //chyba
          $returnmessage = 'Vazbu článek-blok nelze uložit, nebyly
```

```
            vyplněny všechny povinné pole.'.$returnmessage;
        }
}
```

Obrázek 35 – Kontrola, zda došlo k uložení článku do bloku

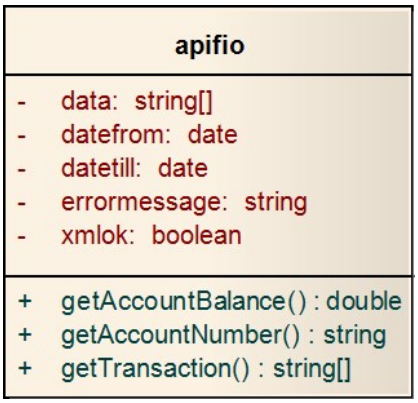
4.9 Bankovní účet

Bankovní účet je důležitá část portálu, neboť díky němu dokáže administrátor zjistit, zda přijmul na svůj účet platbu bez nutnosti se přihlásit na své internetové bankovnictví.

4.9.1 Stav bankovního účtu a platby

O položku v menu, která poskytuje informace o stavu plateb a zůstatcích na účtu, se stará třída *tAccount*. Tato třída je velmi zajímavá, protože pro správnou funkci bankovních výpisů je nutné ji propojit s třídou *apifio*, která byla napsána na základě dokumentace FIO banky. Díky této třídě je umožněno číst data z bankovního účtu banky, u které má administrátor svůj bankovní účet. Bylo nutné na základě uživatelského manuálu a technické dokumentace tyto metody pro výpis dat z účtu napsat a implementovat do systému a vygenerovat tzn. token (náhodný řetězec), který umožnuje vytvořit celou instanci třídy a využívat všechny naprogramované funkce.

Třída *apifio* poskytující metody potřebné ke správné činnosti správy účtu je vidět na obrázku 36.



Obrázek 36 – Třída apifio

Atribut *data* slouží k uložení samotných dat z api, *datefrom* je datum, od kterého se mají transakce zjistit, *datetill* je datum, do kterého chceme znát všechny transakce. Dále jsou v této třídě atributy pro zjištění stavu operace (*xmlok*) a atribut *errormessage*, který slouží k uložení chybové zprávy.

Metody v této třídě jsou pouze tři a to:

a) *getAccontBalance* – tato metoda vrací administrátorovi zůstatek na bankovním účtu,

b) *getAccountNumber* – metoda vrací číslo účtu včetně kódu banky,

c) *getTransaction* – metoda vrací zpracování samotného atributu *data* – seznam transakcí.

Na obrázku 37 je vidět část kódu, umístěná v konstruktoru třídy (speciální metoda, která se volá vždy, kdy je vytvářená nová instance třídy).

```
function __construct($token, $datefrom = null, $datetill = null) {
     if(!$token) return false;
     if($datefrom === null)  $this->datefrom = date("Y-01-01");
     else $this->datefrom = $datefrom;
     if($datetill === null)  $this->datetill = date("Y-m-d");
     else $this->datetill = $datetill;

     // sestavení URL adresy pro přístup k datum FIO
     $url = 'https://www.fio.cz/ib_api/rest/periods/'.$token.'/'.$this-
     >datefrom.'/'.$this->datetill.'/transactions.xml';
     // nutné ošetření přístupu dříve jak 30sec - viz definice API FIO
     $is_free = false;
     if(!isset($_SESSION["lastFioQuery"])) {
          $is_free = true;
          $_SESSION["lastFioQuery"] = time();
     } else {
          if((time() - $_SESSION["lastFioQuery"]) > 30) {
               $is_free = true;
               $_SESSION["lastFioQuery"] = time();
⋮
```

Obrázek 37 – Konstruktor třídy apifio

Tato metoda je v rámci třídy nejdůležitější, neboť získává samotná data v časovém rozmezí a vrací výsledek v atributu *data*.

V první části kódu se ověřuje, zda existuje token, následuje kontrola dat, pokud nejsou data zadána, defaultně se vkládá datum od počátku daného roku a do aktuálního dne. Nutné je dle FIO dokumentace dodržet i minimální časový rámec mezi žádostmi pro práci s účtem, je tedy nutné počkat 30 sekund, než se pošle další požadavek.

Třída *tAccount*, o které se psalo výše, je třída, do které je připojena třída *apifio* a navíc dědí vlastnosti z třídy *class.functions* (viz kapitola 4.1.1.2), protože z této třídy využívá metody pro změnu formátu data. Tato třída zajišťuje přenos dat ze třídy fio banky do naší databáze pomocí metod a umožňuje nastavit filtr dat pro výpis transakcí – metoda *setFilter*.

```
function  setFilter($dates) {
     if($dates["datefrom"] != '') //kontrola prázdného políčka
          $this->datefrom = $this->dateToDB ($dates["datefrom"]);
     if($dates["datetill"] != '')
          $this->datetill = $this->dateToDB ($dates["datetill"]); }
```

Obrázek 38 – Metoda setFilter

4.9.2 Automatické zaznamenávání plateb do systému

Systém má implementovánu metodu, která zajišťuje automatický příjem plateb za semináře, což znamená, že pokud přijde na účet platba a souhlasí variabilní symbol, který systém vygeneroval a odeslal spolu s ostatními daty uživateli na email v rámci pokynů k platbě a souhlasí také částka za seminář, systém sám označí stav platby za zaplacený.

Jak je patrno, toto ověřování plateb musí být spouštěno automaticky. K tomu slouží tzv. cron, což je služba (na serveru – typicky tuto možnost uživatel může využít po zaplacení hostingu), která v určitý čas spustí konkrétní skript, který je dostupný na serveru. Daný skript neřeší samotnou činnost, pouze si hlídá správný čas spuštění daného souboru. Vzhledem k tomu, že tento skript má na starosti označování nezaplacených seminářů statusem „zaplaceno“, musí mít přístup k databázi, kde jsou dostupné informace, jaký kurz byl od jakého uživatele zaplacen. Součástí správné funkcionality musí být i ověření částky za daný seminář.

```
$db = new tDatabase(); //vytvoření objektu pracující s databází
$fio = new apifio(API_TOKEN, date('Y-m-d', strtotime(' -1 day')));
//připojení api banky za aktuální den a den předchozí
```

```
$data = $fio->getTransaction(); //vrátí všechny transakce v daném
období
foreach($data as $item) { //prohledá nezaplacené semináře a porovná VS
     $q  = "SELECT u.sem_price, t.id, t.visit_vs
          FROM seminars_visits t
          INNER JOIN seminars u ON t.visit_sem_id = u.sem_id
          WHERE t.payment_status = '0'
          AND t.visit_vs = '" . $item["vs"] . "'
          ";
     $r  = $db->set_query($q);
     if(count($r) > 0 && $r !== false) {
          if($item["castka"] == $r[0]["sem_price"]) { //ověření částky
               $payment_status = 1;    // 1 = V pořádku zaplaceno
               $payment_note = 'Seminář v pořádku zaplacen.';
          } else {
               $payment_status = -2;   // -2 = Zaplacená špatná částka
               $payment_note = 'Zaplacena částka ' . $item["castka"] .
               'Kč místo ' . $r[0]["sem_price"] . 'Kč.';
          }
          $payment_date   = date("Y-m-d", $item["datum"]); //datum
          platby
          $q = "UPDATE seminars_visits
               SET payment_status = '".$payment_status."', paymentdate
               = '".$payment_date."', `note` = CONCAT(`note`, '\n" .
               $payment_note . "')
               WHERE id = '" . $r[0]["id"] . "'"; //uloží datum platby
               do databáze
          $db->set_query($q);
```

Obrázek 39 – Ukázka kódu souboru, který obsluhuje cron

Tento skript se spouští každý den ve 23:00 (což je nastaveno na hostingu) a je tedy třeba kontrolovat platbu za aktuální i předchozí den. Pokud by byla platba kontrolována pouze za aktuální den, mohl by nastat problém s platbou např. ve 23:30. Systém by tuto platbu nezaznamenal, protože v aktuálním dnu by se spustil ve 23:00 (platba by ještě nebyla připsána na účet), v následujícím dnu se spustí taktéž a prohledá vše od 00:00 do 23:00, tudíž platbu v předešlém dnu (ve 23:30) nezaznamená.

4.10 Partneři a doporučené stránky

Administrátor může prostřednictvím odkazů ve svém menu měnit i partnery a doporučené stránky, které jsou umístěny ve spodní části, konkrétně v zápatí (viz kapitola 3.1.1.6).

4.10.1 Správa partnerů

Sekce partnerů je umístěna v zápatí ihned vedle odkazů vedoucích k přihlášení.

Samotná správa se pak provádí z administračního menu, které je dostupné administrátorovi. Třída, která tuto správu zajišťuje, se nazývá *tPartners*. Není výjimkou, že i tato třída dědí své metody z třídy *class.record* a obsahuje dva atributy – objekty komunikující s databází a se třídou s funkcemi.

Objekt odvozený z této třídy obsahuje funkce pro vytvoření partnera, editaci či smazání partnera. Obsahuje též metodu na vrácení všech dostupných partnerů a také si sama hlídá, odeslání formuláře, aby zajistila správné spuštění funkce.

Na obrázku 40 je zobrazen kód, který realizuje smazání partnera. Je to metoda, které je třeba předat jediný parametr, konkrétně id partnera.

```
function deletePartner($partnerid) {
    $q = "SELECT partner_logo
        FROM partners
        WHERE id = '" . intval($partnerid) . "'"; // dotaz na získání
        informací o názvu obrázku partnera (loga)
    $qr = $this->db->set_query($q); //provedení SQL dotazu
    $filenamepath   = PARTNERS_IMG_PATH.$qr[0]["partner_logo"];
    //cesta k souboru partnera
    if(is_file($filenamepath)) //pokud je dostupný, systém ho smaže
        unlink($filenamepath);
    $q = "DELETE FROM partners
        WHERE id = '" . intval($partnerid) . "'"; //smaže záznam i z
        databáze
    if($this->db->set_query($q))
        return "Partner byl úspěšně smazán z databáze.";
    else
        return "Partnera se <strong>nepodařilo</strong> smazat.<br
```

```
            />Opakujte akci později nebo se obraťte na správce webu.";
}
```

Obrázek 40 – Smazání partnera

Jak je vidět na obrázku 40, funkce pracuje s identifikátorem partnera, podle kterého zjistí název obrázku, potažmo loga. Pokud logo existuje a nedošlo k chybě na serveru, smaže tento obrázek z dané složky a celý záznam vymaže z databáze.

4.10.2 Správa doporučených stránek

Správa doporučených stránek je velmi podobná jako správa partnerů (viz kapitola 4.9.1). Možnost správy doporučených stránek poskytuje třída *tLinks*, která má implementovány metody pro přidání, editaci a smazání doporučených odkazů, samozřejmostí je metody pro získání všech odkazů a taktéž si tento objekt zjišťuje, zda došlo k odeslání formuláře a jaké tlačítko tuto událost vyvolalo, což je potřebná informace pro zavolání správné metody.

Třída taktéž dědí z třídy *class.record* a obsahuje kromě již zmíněných metod i dva parametry. Jeden je objekt pracující s databází, druhý parametr je taktéž objekt, který má přístup k metodám třídy *class.functions*.

```
function editLink($linkid) {
     $q = "SELECT id,name,link,target,position
          FROM links
          WHERE id = '" . intval($linkid) . "'";
     $qr = $this->db->set_query($q); //provedení SQL dotazu
     if(count($qr[0]) > 0 && $qr !== false) //pokud je výsledek dotazu
          $this->setRecord($qr[0]); //uložíme do pole pro další práci
     else
          return false;
}
```

Obrázek 41 – Editace odkazu

Výpis kódu na obrázku 41 popisuje, jak pracuje editace daného odkazu. V první části kódu jsou na základě identifikátoru odkazu zjištěny z databáze všechna data o odkazu, která se vloží do pole.

Administrátorovi je toto pole jinou funkcí zobrazeno a jakákoliv změna v příslušných polích se uloží zpět do databáze.

5 Testování

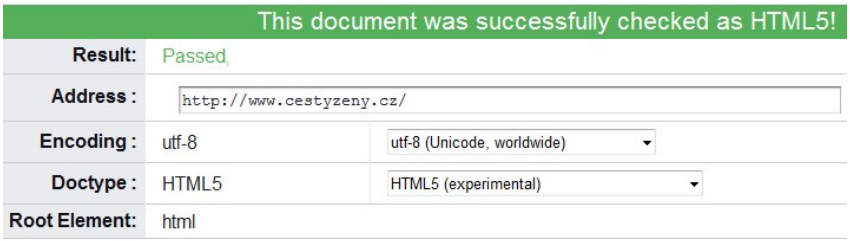
5.1 Testování aplikace

5.1.1 Validita kódu

Validita zdrojového kódu není až tak důležitá pro samotnou funkci portálu, neboť i při určitých nevalidních prvcích bude portál fungovat tak, jak má. O to se stará prohlížeč, který se snaží dodefinovat chybějící atributy a všechny nedostatky určitým způsobem opravit. Různé prohlížeče opravují konkrétní chyby různě, takže se může stát, že prohlížeč opraví kód jinak, než by měl být správně definován a zobrazení na samotné stránce se tudíž může změnit. Validita je dobrá k tomu, aby při nové verzi prohlížeče nemusel programátor přepracovávat části svého kódu podle nových norem.

O validitu se stará mezinárodní konsorcium W3C (World Wide Web Consortium), které vyvíjí spolu s veřejností webové standardy pro weby.

Na stránkách W3C je možnost dané stránky zvalidovat, tj. zjistit, zda daný kód je či není, validní. Uživateli je zobrazen stav (validní/nevalidní). V případě nevalidního kódu je uživatel informován o chybových řádcích pro možnou opravu.



Obrázek 42 – Validita kódu

5.1.2 Použitelnost na více zařízeních

Použitelnost portálu na více zařízení je stále důležitější, neboť většina lidí nepracuje v dnešní době pouze se stolními počítači či notebooky, ale používají mobilní telefony, tablety a další multimediální zařízení, která jsou schopná zobrazit webové stránky. Každé takovéto elektronické zařízení má jinak naprogramované zobrazení, jinou velikost i poměr stran, což může způsobit problémy při správném zobrazování webové stránky. Bylo tedy nutné nakódovat portál tak, aby bylo využito co nejvíce klasických vlastností jazyka CSS, zejména atributy pro velikost, tzn. využívat převážně atributy typu „width:100%“, naopak se vyvarovat použití atributů typu „width:900px“, kde se pevně definuje rozměr. Vzhledem k tomu, že každé zařízení má jiný rozměr

(rozlišení), nelze považovat tento styl kódování za použitelný, je tedy nutné využít přístupu v procentech, kde se z rozměru daného zařízení vezme část, která odpovídá danému procentu, např.

Monitor (1 920 × 1 080 px) při atributu „width: 80%“ se tato hodnota nastaví na 1 536 px.

Notebook (1 366 × 768 px) při atributu „width: 80%“ se tato hodnota nastaví na 1093px.

Je tedy vidět, že pokud se hodnota zadá v procentech, dá se daný problém s různými velikostmi vyřešit a dané bloky pouze přepočítat rozměrově na procenta.

5.2 Testování použitelnosti

Testování samotné použitelnosti, je velice důležitý bod vyjadřující určitou míru kvality v rámci designu i funkcí. Na základě testů dokáže tvůrce portálu vyhodnotit nedostatky či možné vylepšení podle problémů, které vznikali při řešení úkolů uživatelům.

V první řadě je nutné stanovit, jak daný test bude probíhat, přičemž je důležité, aby se samotné testování provádělo až ve finální části portálu, tj. při plné funkčnosti.

5.2.1 Výběr testujících osob

Důležitým bodem testování je výběr testovacích osob, protože na základě jejich výsledků se vytvoří závěr, zda aplikace je použitelná, či je potřeba jí více či méně upravit, co do přehlednosti tak i do úpravy samotných funkcí portálu.

Pro testování portálu bylo vybráno celkem pět osob různého věku, rozlišných znalostí internetu a webových portálů.

5.2.2 Úkoly pro testování

Úkoly pro testování byly určeny na základě funkcí, které portál poskytuje tak, aby byla alespoň z každé sekce funkcí vybrána jedna, kterou je třeba otestovat. Testování proběhlo na základě uživatelského manuálu, který byl za tímto účelem sepsán. Uživatelé se v první části seznámili s portálem, pročetli si uživatelský manuál, který i během testování měli při sobě pro možné získání nápovědy k provedení určité operace.

Uživatelé měli za úkol:

1) Přihlásit se na portál (potřebné údaje budou uživatelům sděleny).

2) Přidat aktualitu s libovolným nadpisem i obsahem.
3) Vytvořit kurz s libovolnými parametry, a zobrazit jej „na hlavní stránce“.
4) Přihlásit se na právě vytvořený seminář a zjistit, jaký je variabilní symbol pro platbu.
5) Vytvořit libovolného partnera nebo doporučený odkaz v zápatí.
6) Editovat uživatelem vytvořený kurz a nastavit cenu o 10,– Kč vyšší (v případě, že editace nepůjde, realizovat příslušnou operaci, aby k editaci mohlo dojít).
7) Okopírovat seminář a změnit libovolné parametry a seminář aktivovat.
8) Smazat vytvořené semináře, aktualitu a odkaz v zápatí.
9) Odhlásit se z portálu.

5.2.3 Výsledky testování

Každý bod testování byl vyhodnocen samostatně.

Uživatelé středního věku v několika málo případech chybovali vlivem nepozornosti, osoby mladší generace či osoby zkušenější s administrací redakčních systémů prošli testem bez obtíží. Na základě pozitivního výsledku testování byl portál nasazen do ostrého provozu.

6 Uvedení do provozu

Uvedení do provozu není prakticky nic jiného než nakopírování celého systému na server a nastavení několika atributů, jako je připojení k databázi, nastavení cron démona, který obstarává platby, viz kapitola 4.9.2.

V první řadě bylo nutné zaregistrovat doménu, což je samotný název – identifikátor webových stránek. Přes tento název se na dané stránky dostanou klienti.

Název byl zvolen jako „cestyzeny.cz“, který dle administrátora popisuje myšlenku celého portálu.

V druhé části se registroval hosting, což je samotný prostor pro data, který portál obsahuje.

Doména a hosting byly registrovány u stejné společnosti, což ulehčuje následnou administraci i případné řešení problémů.

Bylo nezbytné vytvořit vlastní databázi pro ukládání dat, v ní poté příslušné tabulky s patřičnými atributy a do nich importovat důležitá data jako např. názvy stránek, bez kterých by portál nezobrazil žádné stránky ani nadpisy.

7 Závěr

Byl vytvořen webový portál pro správu kurzů, který umožnuje administrátorovi spravovat semináře či aktuality.

V první části práce byla provedena analýza systému v rámci vzhledu, na který byl kladen velký důraz, tak i v rámci požadované funkcionality. Problematika tvorby a návrhu webových portálů byla konzultována s potenciálními zákazníky (zejména kvůli vzhledu a přehlednosti), ale i s profesionálními návrháři webových aplikací. Taktéž byla provedena i rešerše hotových řešení.

Následoval samotný návrh systémů – definice a návrh hlavních tříd portálu včetně popisu jejich funkcionalit.

Na základě návrhu byl celý portál implementován v jazyce PHP.

Pro administrátora byl vytvořen podrobný manuál, před samotným spuštěním portálu byl proveden test na uživatelích a klientech, který byl vyhodnocen jako úspěšný a tím mohl být uveden do provozu.

Daný portál umožňuje spravovat kurzy (vytvářet, mazat, editovat, vytvářet kopie) a aktuality (vytvářet, editovat a mazat).

Dále systém disponuje funkcemi pro správu článků a bloků. Implementovány jsou též metody pro správu uživatelů a správu plateb. Vytvořeno bylo i grafické rozhraní, které veškerá dostupná data uživateli či administrátorovi zobrazí.

Tímto byly splněny všechny body zadání diplomové práce.

8 Použitá literatura a zdroje

[1] BÖTTIGHEIMER, Armin. *Vytváříme grafiku webu ve Photoshopu: průvodce tvorbou ikon, bannerů a navigačních panelů.* Vyd. 1. Brno: Computer Press, 2011, 208 s. ISBN 978-802-5134-856.

[2] CKEditor [online]. 2014 [cit. 2014-03-07]. Dostupné z: http://ckeditor.com/

[3] GILMORE, W. *Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály*. Nové, 3. vyd. Překlad Jan Pokorný. Brno: Zoner Press, 2011, 736 s. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.

[4] CHAFFER, Jonathan. *Mistrovství v jQuery: kompletní průvodce vývojáře*. 1. vyd. Brno: Computer Press, 2013, 384 s. ISBN 978-80-251-4103-8.

[5] *JavaScript - w3school* [online]. 2014 [cit. 2014-03-07]. Dostupné z: www.w3schools.com/js/

[6] *JQuery* [online]. 2014 [cit. 2014-03-07]. Dostupné z: http://jquery.com/

[7] KRUG, Steve. *Nenuťte uživatele přemýšlet!: praktický průvodce testováním a opravou chyb použitelnost [sic] webu.* Vyd. 1. Brno: 2010, 165 s. ISBN 978-80-251-2923-4.

[8] PHP. *PHP* [online]. 2014 [cit. 2014-03-07]. Dostupné z: http://php.net

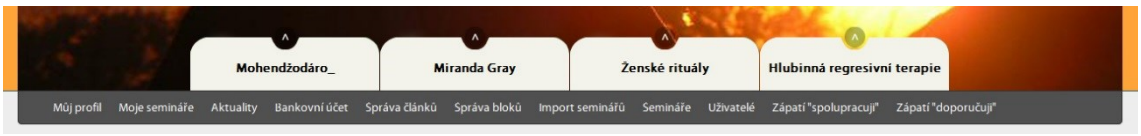
[9] W3C. *CSS3* [online]. 2014 [cit. 2014-03-07]. Dostupné z: http://www.w3schools.com/css

9 Přílohy

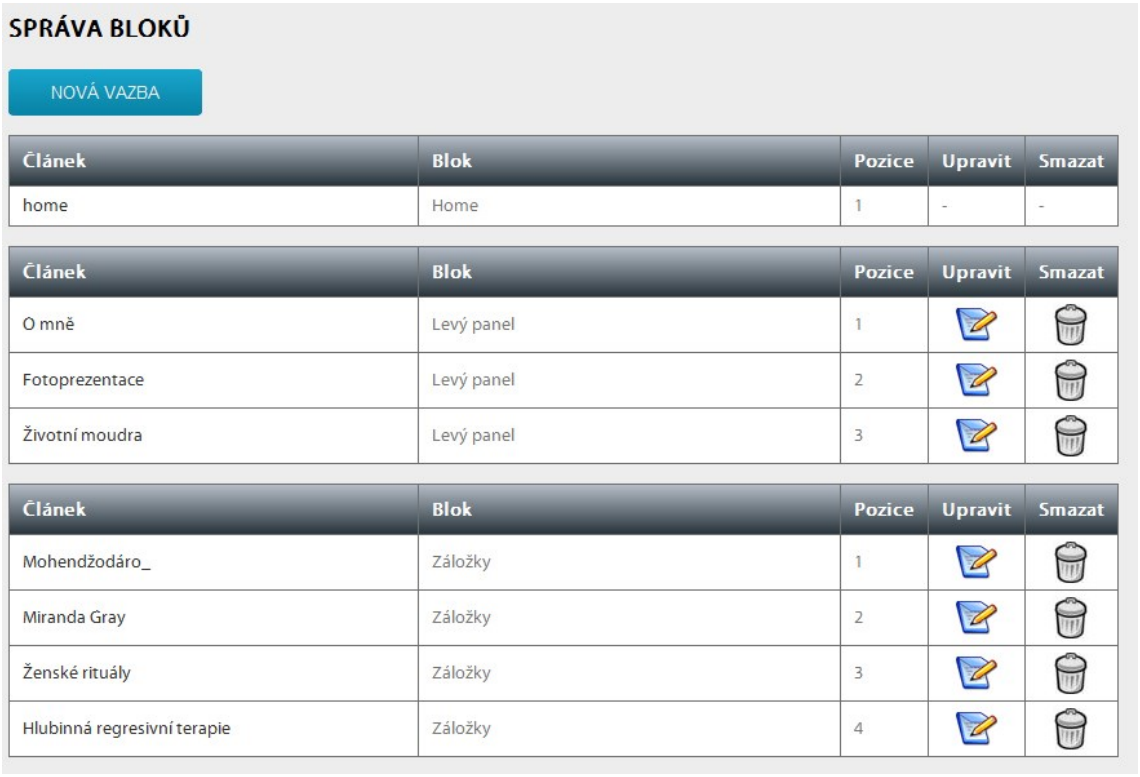
Příloha A – Grafická podoba portálu

Obrázek A.1 – Grafická podoba webového portálu

Obrázek A.2 – Uživatelské menu pro přihlášení



Obrázek A.3 – Administrátorské menu po přihlášení



Obrázek A.4 – Správa bloků pro články

Obrázek A.5 – Semináře konkrétního uživatele

Příloha B – Obsah přiloženého CD

Přiložené CD obsahuje následující adresáře a soubory:

- Webový portál
    - Adresáře
        - *cron* – adresář se skripty, které spouští ze serveru CRON,
        - *css* – adresář obsahující definici stylů pro celý portál,
        - *errorlogs* – složka, ve které se nachází chybové hlášení,
        - *files* – složka obsahující obrázky používající se na portálu,
        - *font* – složka s písmem, které je použito na portálu,
        - *images* – adresář, obsahující všechny grafické prvky portálu,
        - *inc* – adresář, který obsahuje definici hlavních objektů,
        - *js* – soubory potřebné ke správné činnosti grafických prvků,
        - *pages* – adresář se všemi stránkami, které se používají,
        - *partners* – adresář, který obsahuje loga partnerů,
        - *seminars* – adresář s uloženými semináři (obrázky, PDF..).
    - Soubory
        - *core.php* – hlavní soubor pro vytvoření všech objektů,
        - *index.php* – hlavní soubor pro zobrazení celého portálu,
        - *login.php* – soubor k přihlášení uživatelů,
        - *logout.php* – soubor k odhlášení uživatelů,
        - *robots.txt*,
        - *sitemap.xml*,
        - *systemfailed.html* – soubor, který uživateli zobrazí chybu.
- Doplňky
    - Soubory
        - Diplomová práce – Webový portál (PDF),

- *db.sql* – import databáze a všech důležitých záznamů pro potřeby testování či přenesení systému na jiný server,
- Uživatelský manuál (DOCX) – Manuál pro administrátora.